SimData

Nick Miner

November 5, 2018

Question 1

1a

```
N <- 1000
P <- 100
```

1b

```
Sigma = diag(sample(1:10, P, P))
X <- mvrnorm(N, runif(P, -10, 10), Sigma)
#Creating a 100x100 matrix for Sigma with a random, repeat sampling of 1 through 10 running through
#the matrix diagonal. Setting X as a multivariate normal with 100 obs, a uniform mean of 0, and Sigma
#as the covariance matrix for the data.
```

1c

```
p <- rbinom(P, 1, 0.1)
#vector p from Bernoulli distribution.
```

1d

```
beta = p * rnorm(P, 5, sqrt(5)) + (1 - p) * rnorm(P, 0, sqrt(0.1))
epsilon <- rnorm(N, 0, sqrt(10))
y <- X %*% beta + epsilon
#Creating beta to draw from different normal distrubtions based on if p is 1 or 0. Setting epsilon
#as a normal distribution with mean 0 and sd of sqrt(10). Creating y as a function of X times beta
#plus error term epsilon.
```

1e

```
dataset1 <- as.data.frame(cbind(y,X))
names(dataset1) <- c("y", rep(paste("X", 1:ncol(X), sep = "")))
train.index <- sample(1:1000, size = 800, replace = FALSE)
test.index <- setdiff(1:1000, train.index)
training.set <- dataset1[train.index,]
test.set <- dataset1[test.index,]
#Creating a dataset from x and Y, then setting the colnames for the dataset. Creating training and
#test sets from a random sampling of the data without overlap.
```

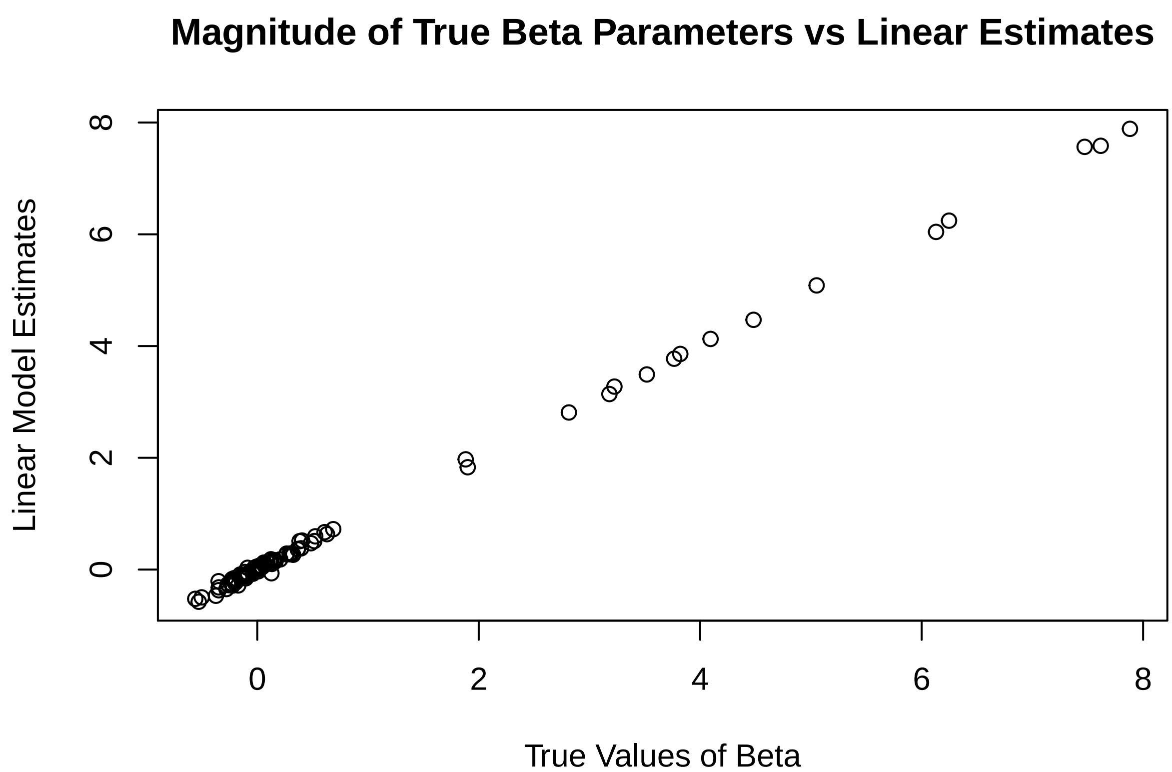
Question 2

2a

```r
q2.lm <- lm(y ~ X)
coef(summary(q2.lm))[1:20,]
```

```
##                 Estimate Std. Error      t value      Pr(>|t|)
## (Intercept)  0.050508616 3.34853995   0.01508377  9.879687e-01
## X1           0.112978844 0.03829196   2.95045898  3.255360e-03
## X2          -0.278295706 0.03492325  -7.96877992  4.838691e-15
## X3          -0.240938027 0.03870069  -6.22567818  7.348743e-10
## X4          -0.156600438 0.03539987  -4.42375719  1.088805e-05
## X5           0.121391724 0.04971833   2.44158894  1.481466e-02
## X6          -0.286864608 0.07535648  -3.80676773  1.503404e-04
## X7           0.031884159 0.05450956   0.58492778  5.587430e-01
## X8           0.722684497 0.04271400  16.91914912  5.862950e-56
## X9          -0.008950315 0.03836953  -0.23326622  8.156078e-01
## X10         -0.070981390 0.03591945  -1.97612704  4.844575e-02
## X11         -0.348579149 0.10606934  -3.28633269  1.054317e-03
## X12         -0.193412405 0.07535315  -2.56674615  1.042662e-02
## X13          0.163947684 0.03701727   4.42895138  1.063460e-05
## X14         -0.065662045 0.10425569  -0.62981736  5.289741e-01
## X15          0.044494810 0.05442269   0.81757825  4.138147e-01
## X16          1.831336774 0.04816250  38.02412041 2.394697e-189
## X17          0.124612878 0.05335538   2.33552621  1.973507e-02
## X18          7.583933312 0.04788321 158.38398492  0.000000e+00
## X19         -0.575859709 0.07589338  -7.58774599  8.127662e-14
```

```r
#Fitting linear model and examining the first 20 coefficients.


#Plotting our data generated beta values against linear model
#estimates. Plotting the data shows a perfect match of the linear
#estimates tothe true values of beta.
plot(beta, coef(q2.lm)[-1], main = "Magnitude of True Beta Parameters vs Linear Estimates",
     xlab = "True Values of Beta", ylab = "Linear Model Estimates")
```



2b

```
#To test the predictive accuracy of the model, I fit the linear model to my previously created
#training dataset. Then I found the mse of that model to be 8.93. I then fit the training model to
#the previously created test set to create test predictions. I found the mse of the test predictions
#to be 12.21. The training and test mse are reasonably close and reasonably low. Still, a difference
#of over 3 seems to indicate the model might not be the best fit.

training.model <- lm(y~., data = training.set)
training.mse <- mean(((training.set$y - training.model$fitted.values)^2))
training.mse
```

```
## [1] 8.934155
```

```
#setting a linear model to the previously created training data. Then setting and viewing the mean
#squared error of the test set as a manner for assessing model predictive accuracy.

test.preds <- predict(training.model, newdata = test.set)
test.mse <- mean(((test.set$y - test.preds)^2))
test.mse
```

```
## [1] 12.2089
```

```
#Predicting values for the test set using the model fit on the training set. Then finding the mean
#squared error of the test predictions.
```

Question 3

3a

```
#Penalized regression is used in cases where the number of predictors p > the number of obs n.
#Penalized regression introduces some form of penalty based on the beta coefficients, multiplied
#by a regularization parameter lambda that controls the penalty vs model fit. This penalty shrinks
#beta coefficients towards zero, thus constraining the abundance of parameters and allowing for a
#model to be fit to the data.
```

3b

```
#In a LASSO regression, the regression penalty is the sum of the absolute value of Beta_j. Multiplied
#by regulator lambda, this regression approaches the target point by setting the mean of the
#distribution at 0 and narrowing the parameters diagonally. LASSO regressions don't have as many nice
#features as ridge regressions, but shrinks many parameters to exactly 0, which can lead to cleaner
#results.

#For a ridge regression the penalty is the sum of squared Beta_j multiplied by regulator lambda.
#Ridge regression sets the mean of the distribution at 1 and approaches the target point circularly.
#Ridge regressions are useful for data with multicollinearity, optimize more easily, and are able to
#be smoothed for stable results.

#Lastly, an Elastic Net regression uses a combination of both Ridge and LASSO. Both penalties are
#multiplied by the same lambda, but then also multiplied by an alpha value between 0 and 1. The sum
#of the penalty values selected for by one value of alpha are added to the sum of the values for the
#opposing regression multiplied by 1 - alpha to ensure independent draws. Elastic Net regressions
#linearly combine the Ridge and LASSO regressions for the most efficient way to constrain parameters.
#The Ridge Regression shrinks most parameters close to zero while the LASSO Regression shrinks most
#parameters to zero. The Elastic Net draws from each based on probability selector alpha, and then
#sums the values produced by both regressions selected by alpha. Because two penalties are applied,
#there can be "double-shrinkage"; ie the model may have shrank the selection area too quickly and
#missed the target value. To combat this the Elastic Net Regression uses a quadratic penalty. The
#Elastic Net regression combines the best features of LASSO and Ridge regressions while also
#accounting for new penalties more effictively.
```

3c

```
#My own research was to examine the combat effictiveness of Marines in Iraq from 2001-2009. However,
#I would have to account for numerous parameters, such as time of year, weather, number of combat
#veterans per squad, number of new recruits per squad, gear variations, etc. Since my data would be
#coming from the DOD, there is a strong chance my parameters would far exceed my observations. A
#linear model would be extremely ineffective, if not entirely useless. A penalized regression model,
#however, may help to control the parameters so effective results can be observed.
```

Question 4

4a

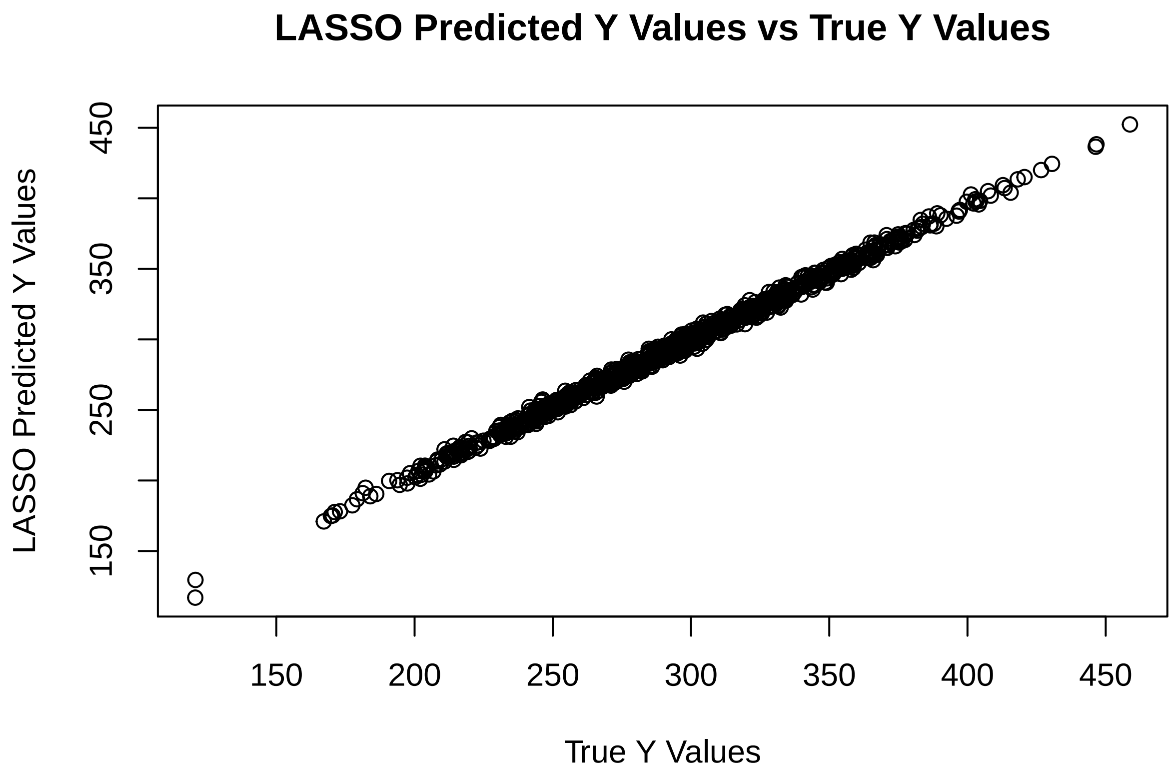
```r
mod.lasso = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 0, lambda = seq(0, 50, .5)),
                  data = dataset1, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                      number = 2,
                                                                                      search = "grid"))
#fitting LASSO model with broad search spectrum
mod.lasso$bestTune
```

```
##   alpha lambda
## 5     0      2
```

```r
#best fit alpha and lambda for search area

mod.lasso2 = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 0, lambda = seq(1, 3, .1)),
                  data = dataset1, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                      number = 2,
                                                                                      search = "grid"))
#re-running LASSO with lambda parameters reset to above and below lambda value found from first run.
mod.lasso2$bestTune
```

```
##    alpha lambda
## 12     0    2.1
```

```r
#showing new alpha and lambda

#Creating predictions for y based off of second LASSO run, then plotting those y values vs the data
#generated y values. The plot shows the predicted y values and actual y values to be an extremely
#close fit, with only a few outliers. The mse for the model is calulated as 14.40.
lasso.y.preds <- predict(mod.lasso2)
plot(dataset1$y, lasso.y.preds, main = "LASSO Predicted Y Values vs True Y Values",
     xlab = "True Y Values", ylab = "LASSO Predicted Y Values")
```




```
lasso.mse <- mean(((dataset1$y - lasso.y.preds)^2))
lasso.mse
```

```
## [1] 14.39556
```

4b

```
mod.ridge = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 1, lambda = seq(0, 50, .5)),
                  data = dataset1, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                       number = 2,
                                                                                       search = "grid"))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
#setting broad ridge
mod.ridge$bestTune
```

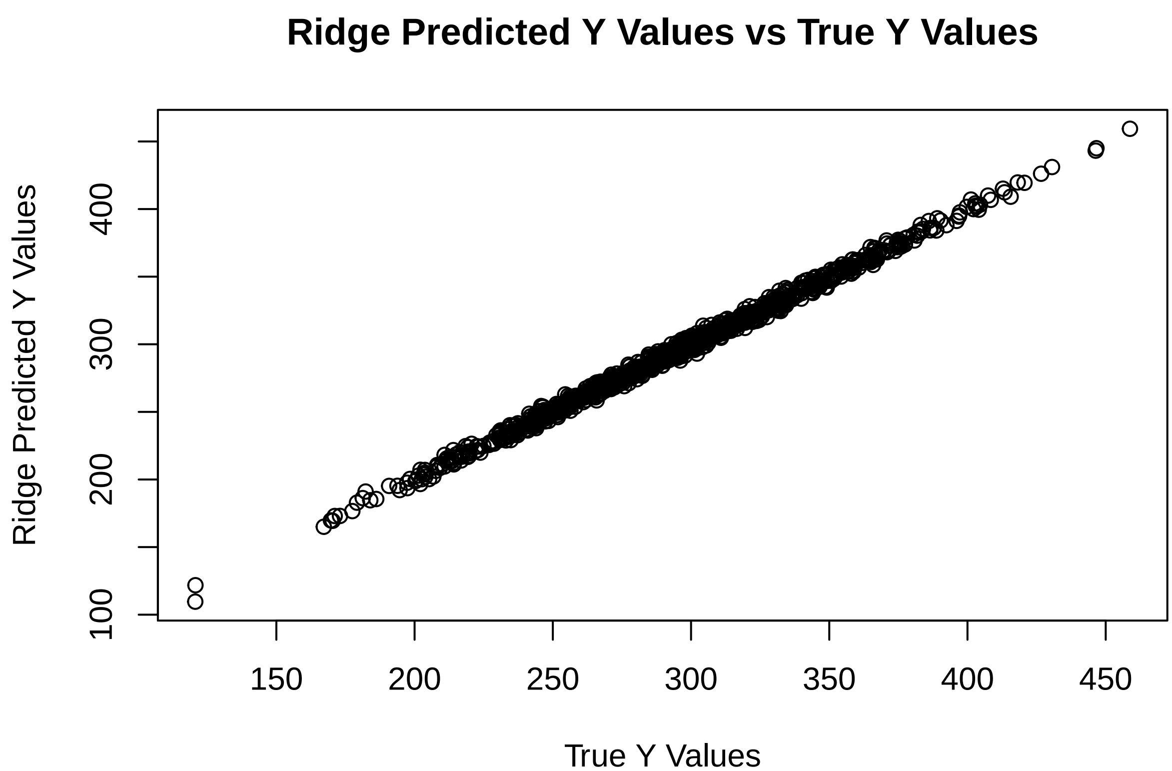
```
##   alpha lambda
## 1     1      0
```

```
#showing alpha and lambda

mod.ridge2 = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 1, lambda = seq(-1, 1, .1)),
                  data = dataset1, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                       number = 2,
                                                                                       search = "grid"))
```

```r
#second ridge run with new parameters
mod.ridge2$bestTune
```

```
##    alpha lambda
## 11     1      0
```

```r
#new alpha and lambda

#Creating y predictions based on the second ridge model and then plotting those predictions vs the
#data generated y values. The Ridge regression shows a model fit as least as tight as the LASSO, and
#actually has a lower mse: 9.34 vs 14.40.
ridge.y.preds <- predict(mod.ridge2)
plot(dataset1$y, ridge.y.preds, main = "Ridge Predicted Y Values vs True Y Values",
     xlab = "True Y Values", ylab = "Ridge Predicted Y Values")
```




```r
ridge.mse <- mean(((dataset1$y - ridge.y.preds)^2))
ridge.mse
```

```
## [1] 9.340122
```

4c

```r
mod.elastic.net = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = .5,
                                                                     lambda = seq(0, 50, 1)),
                    data = dataset1, preProcess = c("center"), trControl = trainControl(method = "cv",
```

```r
                                                                                          number = 2,
                                                                                          search = "grid"))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

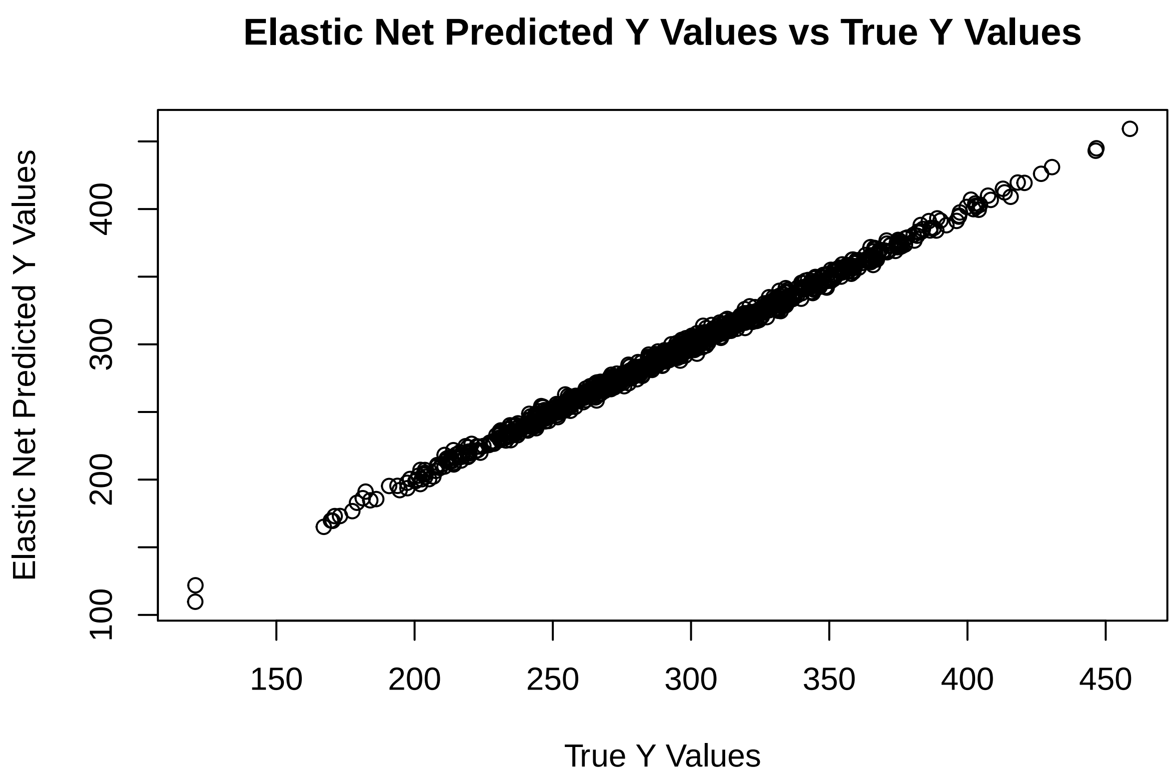
```r
#Fitting elastic net model. Alpha is set as .5 for even selection from Ridge and LASSO penalties.
mod.elastic.net$bestTune
```

```
##   alpha lambda
## 1   0.5      0
```

```r
#finding best fit alpha and lambda.

mod.elastic.net2 = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = .5,
                                                                       lambda = seq(0, 2, .1)),
                  data = dataset1, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                          number = 2,
                                                                                          search = "grid"))
#running elastic net second time.
mod.elastic.net2$bestTune
```

```
##   alpha lambda
## 1   0.5      0
```

```r
#new alpha and lambda

#Generating y predictions based off of the second elastic net run and plotting those values agains the
#data generated y values. This plot shows the model fits even more tightly than the LASSO and Ridge
#regressions. The model also has one of the lowest mse's  at 9.35.
elastic.net.y.preds <- predict(mod.elastic.net2)
plot(dataset1$y, elastic.net.y.preds, main = "Elastic Net Predicted Y Values vs True Y Values",
     xlab = "True Y Values", ylab = "Elastic Net Predicted Y Values")
```




```r
elastic.net.mse <- mean(((dataset1$y - elastic.net.y.preds)^2))
elastic.net.mse
```

```
## [1] 9.350061
```

4d

```r
#All 3 of the models performed far superiorly to the linear model. Because the models are designed
#to penalize additional betas produced by parameters when p > n, the models constrain the parameters
#to allow for regression in a way that linear regressions do not. Essentially penalized regressions
#constrain excess parameters to allow for linear regression through the penalties.

#The best model appears to be the Ridge regression, in terms of predicted y values to DGP y values.
#The LASSO model performs the worst, and the Elastic Net regression performs only slightly worse than
#the Ridge regression. Because the Elastic Net regression utilizes the LASSO penalty in part, and the
#LASSO model performs poorly on collinear data, it appears that this data is somewhat collinear.
```

Question 5

5a

```r
#The linear model should not work in this case. Since the number of predictors p is larger than the
#number of observations n there may be an infinite number of modeling options for the model. This
```

```r
#also means there is no longer a unique least-squares coefficient estimate since variance for data
#points is infinite. Below is a repeat of the data generating process from question 2, but with p =
#1500.

N <- 1000
P <- 1500
Sigma = diag(sample(1:10, P, P))
X <- mvrnorm(N, runif(P, -10, 10), Sigma)
p <- rbinom(P, 1, 0.1)
beta = p * rnorm(P, 5, sqrt(5)) + (1 - p) * rnorm(P, 0, sqrt(0.1))
epsilon <- rnorm(N, 0, sqrt(10))
y <- X %*% beta + epsilon

dataset2 <- as.data.frame(cbind(y,X))
names(dataset2) <- c("y", rep(paste("X", 1:ncol(X), sep = "")))
train.index2 <- sample(1:1000, size = 800, replace = FALSE)
test.index2 <- setdiff(1:1000, train.index2)
training.set2 <- dataset2[train.index2,]
test.set2 <- dataset2[test.index2,]
```

5b

```r
#The summary of the linear model reports that all 1000 residuals are 0 with 0 degrees of freedom. While
#the variable coefficients have values, there is no standard error, t-values or probabilities reported
#for any variables. There are no residuals, degrees of freedom or anything other than coefficient values
#because the number of parameters forced the data into higher dimensions, resulting in the data being
#too sparse to attribute probabilities correctly to the model.

q5.lm <- lm(y~X)
coef(summary(q5.lm))[1:20,]
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15271.718150        NaN     NaN      NaN
## X1           -113.434175        NaN     NaN      NaN
## X2              9.804141        NaN     NaN      NaN
## X3            143.917326        NaN     NaN      NaN
## X4            -11.386549        NaN     NaN      NaN
## X5           -244.845930        NaN     NaN      NaN
## X6             75.448846        NaN     NaN      NaN
## X7            251.931083        NaN     NaN      NaN
## X8             11.260908        NaN     NaN      NaN
## X9            173.432319        NaN     NaN      NaN
## X10            44.952362        NaN     NaN      NaN
## X11            -5.501978        NaN     NaN      NaN
## X12            19.846197        NaN     NaN      NaN
## X13           -49.382156        NaN     NaN      NaN
## X14          -411.450426        NaN     NaN      NaN
## X15            70.578495        NaN     NaN      NaN
## X16          -157.511800        NaN     NaN      NaN
## X17           -18.856145        NaN     NaN      NaN
## X18            28.924985        NaN     NaN      NaN
## X19            76.399507        NaN     NaN      NaN
```

5c

5c-LASSO

```r
q5.cv.lasso <- cv.glmnet(x = X, y = y, alpha = 0)
#using cross-validation to find the optimal starting lambda value for LASSO.

q5.cv.lasso$lambda.min
```

```
## [1] 391.6764
```

```r
q5.cv.lasso$lambda.1se
```

```
## [1] 471.7756
```

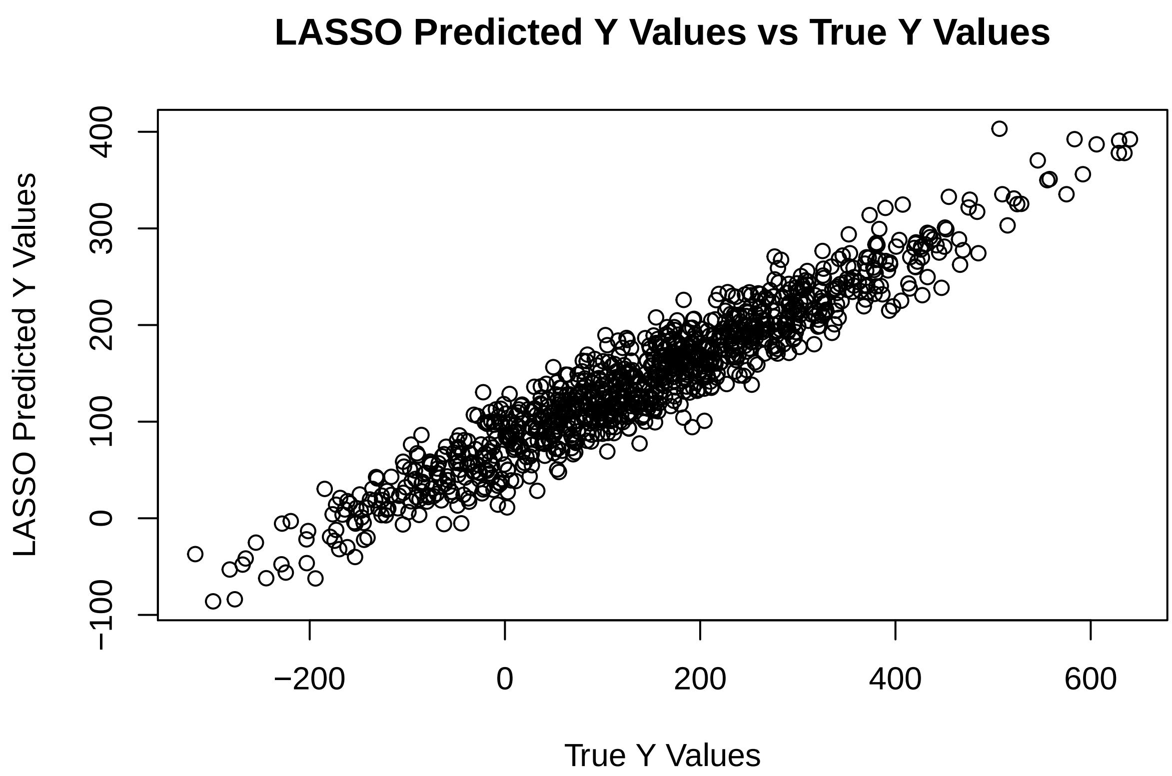
```r
#Finding the minimum lambda value and 1 se value.

q5.mod.lasso = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 0, lambda = seq(0, 6, .1)),
                  data = dataset2, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                       number = 2,
                                                                                       search = "grid"))
#Setting parameters for lambda as the value of 1 se above and below the minimum lambda value, rounded
#to the nearest tenth.
q5.mod.lasso$bestTune
```

```
##    alpha lambda
## 61     0      6
```

```r
#new alpha and lambda

q5.mod.lasso2 = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 0,
                                                                     lambda = seq(3, 9, .001)),
                  data = dataset2, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                       number = 2,
                                                                                       search = "grid"))
#setting new lambda and search as dictated by 1 se.
q5.mod.lasso2$bestTune
```

```
##      alpha lambda
## 6001     0      9
```

```r
#new alpha and lambda.

#Predicting y values determined by the second LASSO run, then plotting those values against the data
#generated y values. The plot shows the model does fit, but not nearly as neatly as in the previous
#question. In addition, the mse is 7129.41, which is incredibly high compared to the previous models'
#mse. The LASSO model's poor fit to the data might indicate that the data is multicollinear, and
#therefore the LASSO is ignoring the data clustering when it draws.

q5.lasso.y.preds <- predict(q5.mod.lasso2)
plot(dataset2$y, q5.lasso.y.preds, main = "LASSO Predicted Y Values vs True Y Values",
     xlab = "True Y Values", ylab = "LASSO Predicted Y Values")
```




```
q5.lasso.mse <- mean(((dataset2$y - q5.lasso.y.preds)^2))
q5.lasso.mse
```

```
## [1] 7514.088
```

5c-Ridge

```
q5.cv.ridge <- cv.glmnet(x = X, y = y, alpha = 1)
#using cross-validation to determine optimal starting lambda value

q5.cv.ridge$lambda.min
```

```
## [1] 0.4103264
```

```
q5.cv.ridge$lambda.1se
```

```
## [1] 0.4942395
```

```
#finding the minimum lambda value and the value of 1 se for the model.

q5.mod.ridge = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 1,
                                                                   lambda = seq(-0.05, 0.2, .001)),
                  data = dataset2, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                       number = 2,
                                                                                       search = "grid"))
#Ridge regression with starting lambda search.
```

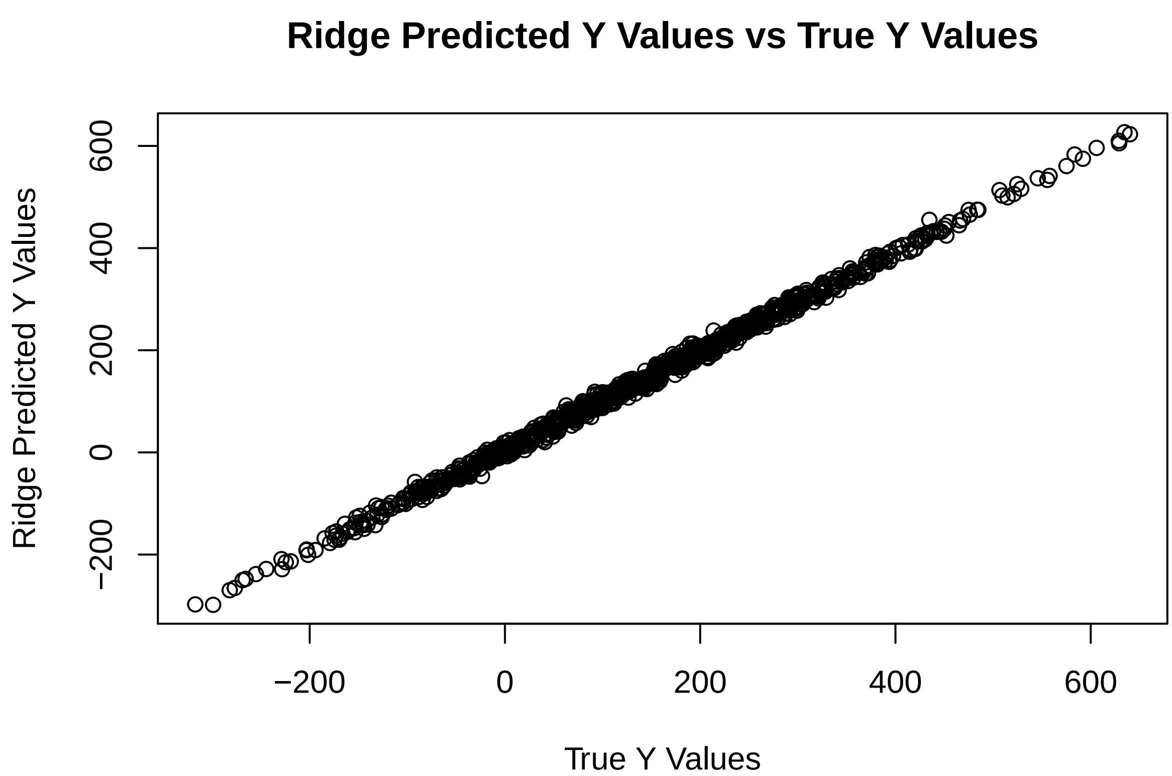
```r
q5.mod.ridge$bestTune
```

```
##     alpha lambda
## 251     1    0.2
```

```r
#new alpha and lambda

q5.mod.ridge2 = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 1,
                                                                    lambda = seq(0.1, 0.3, 0.01)),
                  data = dataset2, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                       number = 2,
                                                                                       search = "grid"))
#second ridge run with new lambda parameters.
q5.mod.ridge2$bestTune
```

```
##    alpha lambda
## 21     1    0.3
```

```r
#new alpha and lambda.

#This plot shows the ridge model fits the data much more neatly than the LASSO model. Also, the mse
#of the model is 80.23, which is significantly smaller than the LASSO's mse.

q5.ridge.y.preds <- predict(q5.mod.ridge2)
plot(dataset2$y, q5.ridge.y.preds, main = "Ridge Predicted Y Values vs True Y Values",
     xlab = "True Y Values", ylab = "Ridge Predicted Y Values")
```



```r
q5.ridge.mse <- mean(((dataset2$y - q5.ridge.y.preds)^2))
q5.ridge.mse
```

```
## [1] 100.4291
```

5c-ENet

```r
q5.cv.enet <- cv.glmnet(x = X, y = y, alpha = 0.5)
#Using cross-validation to discover starting lambda

q5.cv.enet$lambda.min
```

```
## [1] 0.7477482
```

```r
q5.cv.enet$lambda.1se
```

```
## [1] 0.8206527
```

```r
#minimum lambda value and the value for 1 se for the model.

q5.mod.enet = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = 0.5,
                                                                    lambda = seq(-0.08, 0.3, 0.01)),
                    data = dataset2, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                          number = 2,
                                                                                          search = "grid"))
#Fitting Elastic Net model to data with search parameters around starting lambda.

q5.mod.enet$bestTune
```

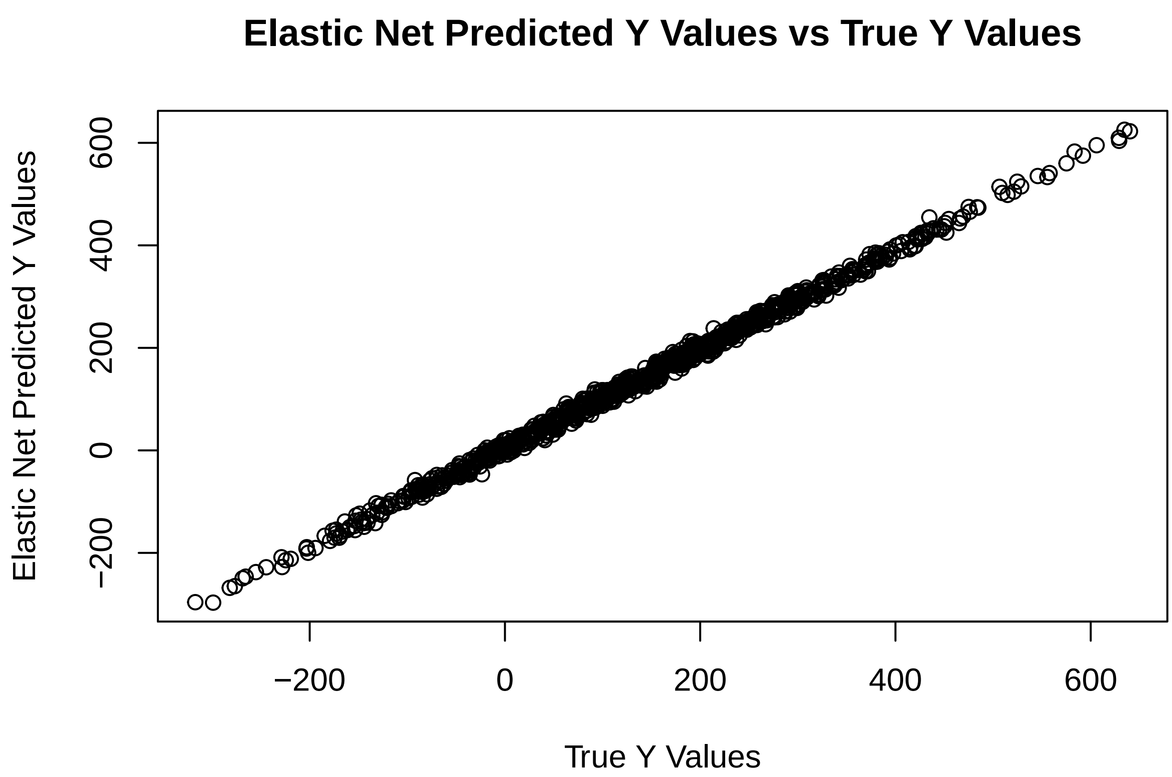
```
##    alpha lambda
## 39   0.5    0.3
```

```r
#new alpha and lambda

q5.mod.enet2 = train(y~., method = "glmnet", tuneGrid = expand.grid(alpha = .5,
                                                                     lambda = seq(0.2, 0.4, .001)),
                    data = dataset2, preProcess = c("center"), trControl = trainControl(method = "cv",
                                                                                          number = 2,
                                                                                          search = "grid"))
#New Elastic Net model with lambda parameters set around new value.

q5.mod.enet2$bestTune
```

```
##     alpha lambda
## 201   0.5    0.4
```

```r
#new alpha and lambda

#The plot shows this model also fits the data much more neatly than the LASSO model, although not quite
#as neatly as the Ridge model. Since the Elastic Net model relies on penalties from the other 2 models,
#the penalties from the poorly-fit LASSO model may be causing some differences in the beta coefficients
#causing the Elastic Net model to be a slightly poorer fit than just the Ridge model. The mse for the
#Elastic Net is 83.11, which is only slightly higher than the Ridge model's. Elastic Net models also
#work better with a larger number of observations, so 1000 obs may not have been enough.

q5.enet.y.preds <- predict(q5.mod.enet2)
```

```r
plot(dataset2$y, q5.enet.y.preds, main = "Elastic Net Predicted Y Values vs True Y Values",
     xlab = "True Y Values", ylab = "Elastic Net Predicted Y Values")
```



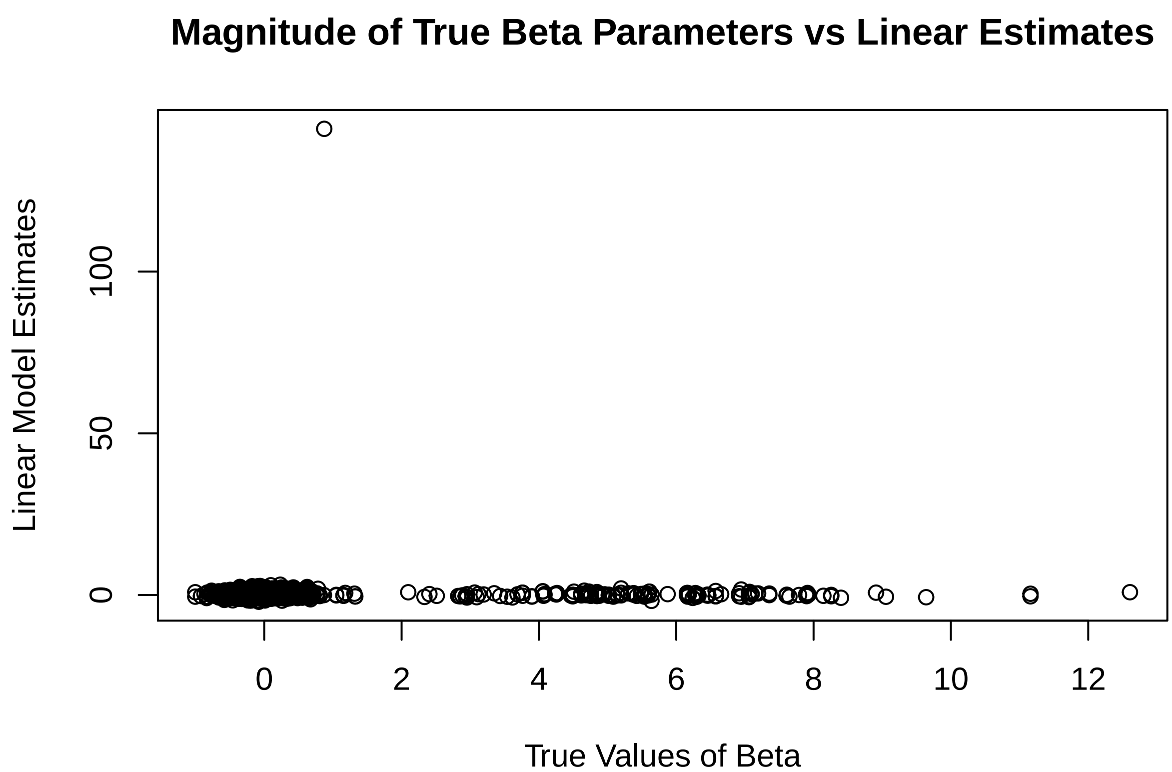

```r
q5.enet.mse <- mean(((dataset2$y - q5.enet.y.preds)^2))
q5.enet.mse
```

```
## [1] 104.1902
```

5d

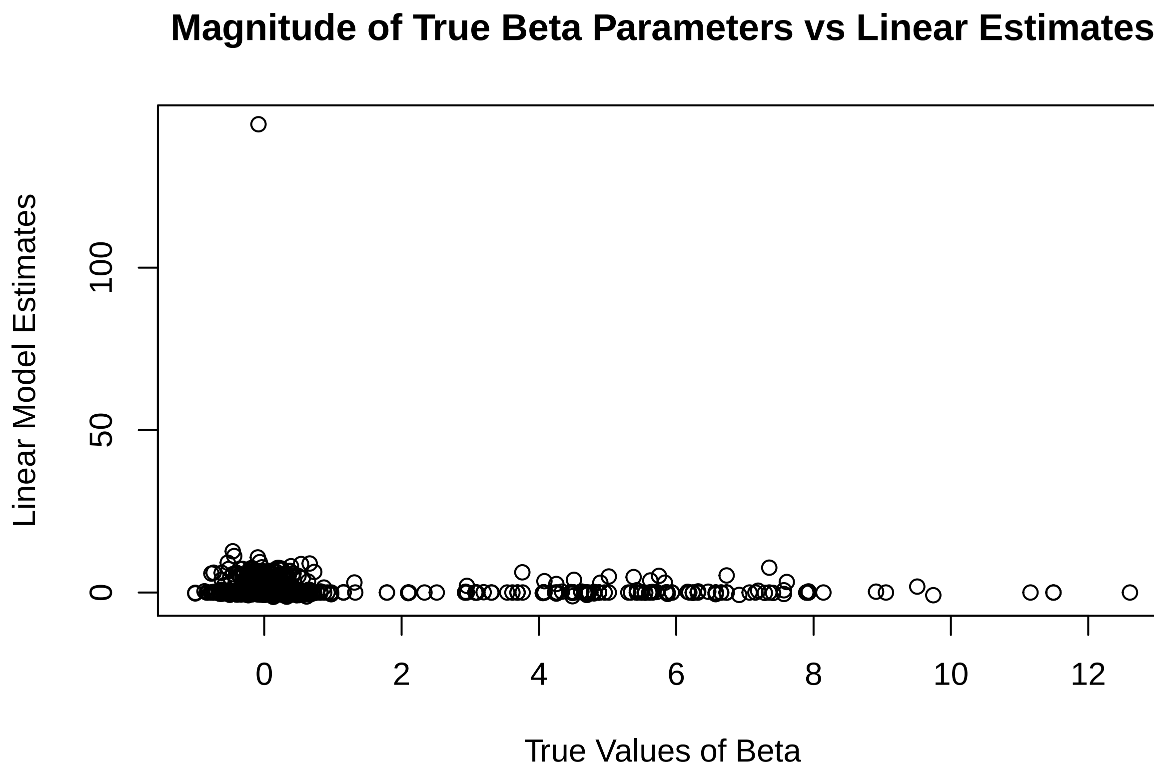
5d-LASSO

```r
#The predicted y values vs data generated y values were examined in question 5c, so in question 5d I
#will examine the true beta values vs the model estimates.

#The beta estimates from the LASSO model plotted against the DGP beta values shows the model is an
#excellent fit for the data. There is only one extreme outlier, while the rest of the points lie
#along the axis. The plot does show many points clustered around (0,0), indicating possible
#multicollinearity in the data.

plot(sample(beta, size = length(coef(q5.mod.lasso2$finalModel, q5.mod.lasso2$bestTune$lambda)),
            replace = TRUE), coef(q5.mod.lasso2$finalModel, q5.mod.lasso2$bestTune$lambda),
     main = "Magnitude of True Beta Parameters vs Linear Estimates",
     xlab = "True Values of Beta", ylab = "Linear Model Estimates")
```



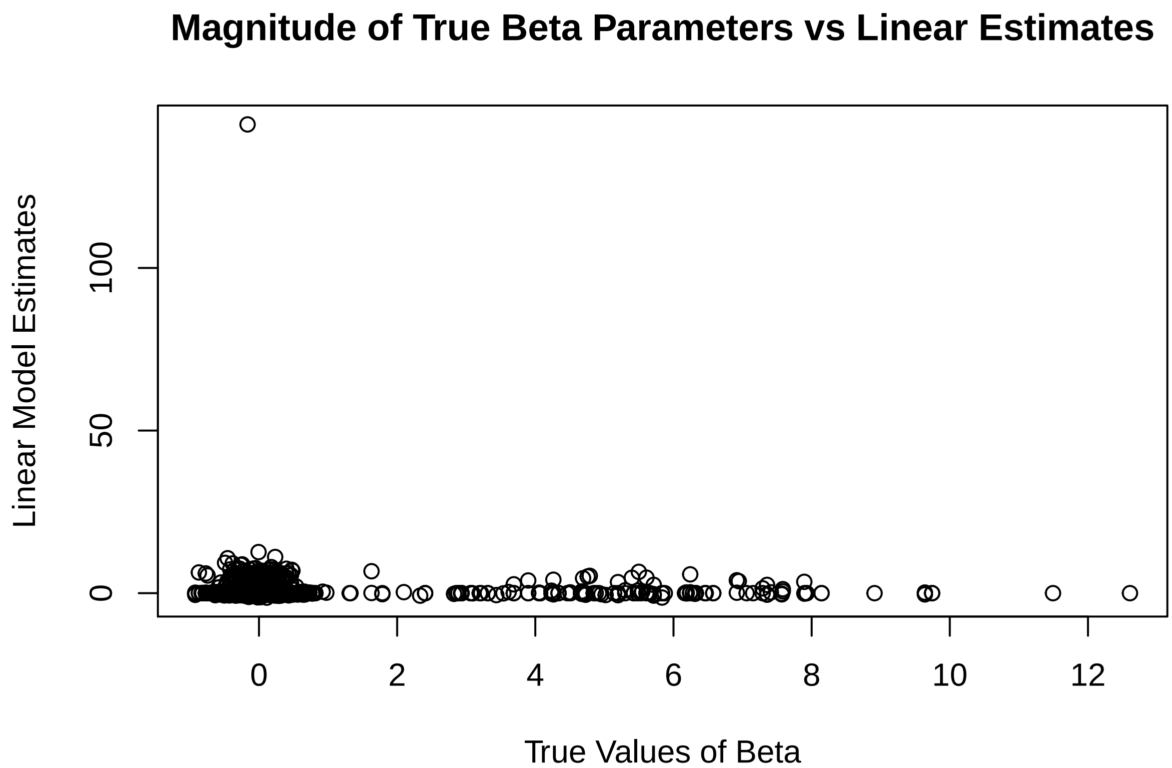
5d-Ridge

```
#The ridge plot is very similar to the LASSO plot. There is only one extreme outlier and much of the
#data appears clustered around (0,0).

plot(sample(beta, size = length(coef(q5.mod.ridge2$finalModel, q5.mod.ridge2$bestTune$lambda)),
            replace = TRUE), coef(q5.mod.ridge2$finalModel, q5.mod.ridge2$bestTune$lambda),
     main = "Magnitude of True Beta Parameters vs Linear Estimates",
     xlab = "True Values of Beta", ylab = "Linear Model Estimates")
```



5d-ENet

```
#The Elastic Net follows the same pattern as the other plots, with one extreme outlier and much of
#the data centered on (0,0).

plot(sample(beta, size = length(coef(q5.mod.enet2$finalModel, q5.mod.enet2$bestTune$lambda)),
            replace = TRUE), coef(q5.mod.enet2$finalModel, q5.mod.enet2$bestTune$lambda),
     main = "Magnitude of True Beta Parameters vs Linear Estimates",
     xlab = "True Values of Beta", ylab = "Linear Model Estimates")
```



5e

```
#For this procedure, I had a dataset consisting of 1000 observations n and 1500 predictors P. The data
#generating process was comprised from Y = X*Beta + Epsilon. Y is the resulting dataframe of
#multivariate matrix X (drawn from a uniform distribution and multiplied diagonally against matrix Sigm
#multiplied by Beta (drawn from 2 different normal distributions based on propensity p) with an added
#error term (drawn from a normal distribution).

#I divided the resulting dataframe Y into training and test sets. I applied a linear model to the data,
#but the linear model reported no results. Because the number of predictors P outnumbers the number
#of obs n, there may be an infinite number of modeling options for the model. This also means there
#is no longer a unique least-squares coefficient estimate since variance for data points is infinite.

#Afterwards I applied  LASSO, Ridge, and Elastic Net regression models to the data. This was necessary
#because the aforementioned models apply a penalized regression to the data. This penalized regression
#helps to constrain additional parameters to allow for linear regression to be accomplished. I began
#with the LASSO model, which produced predicted y values close to the DGP y values. However, the
#SE of the model was incredibly high, and the y value and beta value plotting indicated another model
#may perform better. The Ridge model fit the data much more tightly in the y and beta values plots,
#and had a significantly smaller SE. Finally, the Elastic Net model performed better than the LASSO
#model, but not quite as well as the Ridge regression. The LASSO regression deals poorly with
#collinearity, ignoring data clustering on its draws. Since the LASSO regression and Elastic Net
#regression (which uses the LASSO penalty in part) both performed more poorly on the data than the
#Ridge regression, it seems the data is in some part collinear.
```

```
#Improving on this model would include finding a penalized regression technique that performs well
#on collinear data. A regression only using that model could be performed, and then the Elastic Net
#regression could be re-used with that regression method's penalty replacing the LASSO penalty.
#Then another comparison could be drawn between the models to evaluate accuracy.
```